# CSC 446, HW#3, Kefu Zhu

## 1. Use of `dev` dataset

In this assignment, I used the `dev` dataset to tune the values of the following hyperparameters

- Number of iterations
- Learning Rate
- Number of neurons in the single hidden layer

Based on the mechanics of SGD, I choose to first use a larger learning rate to find the proper range for the number of iterations and drastically decrease the loss to a reasonable range.

And then continue training the current model while performing a grid search for

- Different value of smaller learning rate: $lr = [0.01, 0.5, 1, 2]$
- Different number of neurons in the hidden layer: $hidden\_dim = [3, 5, 25]$
- Number of additional iterations: $iterations = range(0, 200)$

## 2. Experiment

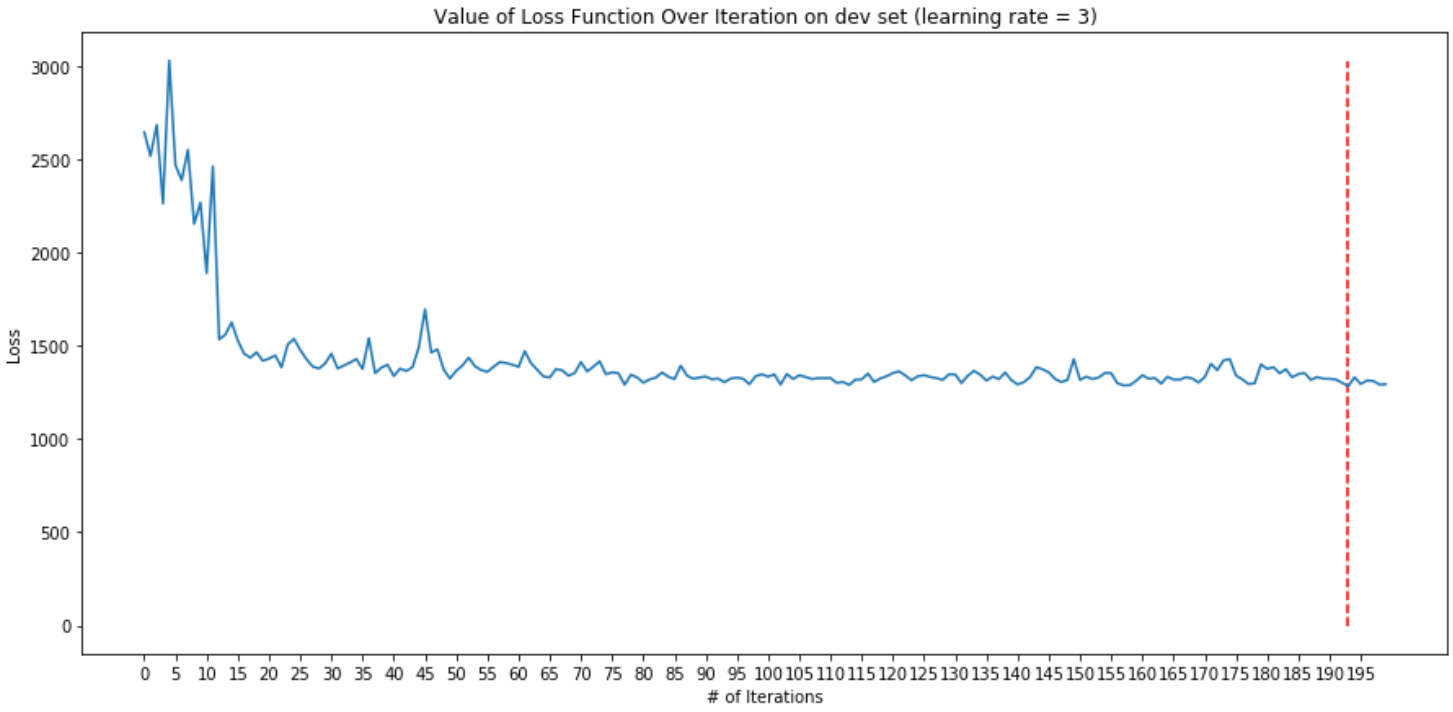
In order to fix the initialization of matrix, I added an additional seed parameter to the `args` and also modified the `init_model` for the Stage 1 of my experiment (use large learning rate to find proper number of iterations).

```
1 | weights_group.add_argument('--seed', type=int, default=123, help='Seed for randomi
```

```python
def init_model(args):
    w1 = None
    w2 = None

    if args.weights_files:
        with open(args.weights_files[0], 'r') as f1:
            w1 = np.loadtxt(f1)
        with open(args.weights_files[1], 'r') as f2:
            w2 = np.loadtxt(f2)
            w2 = w2.reshape(1,len(w2))
    else:
        #TODO (optional): If you want, you can experiment with a different random
        np.random.seed(args.seed)
        w1 = np.random.rand(args.hidden_dim, NUM_FEATURES) #bias included in NUM_F
        w2 = np.random.rand(1, args.hidden_dim + 1) #add bias column

    #At this point, w1 has shape (hidden_dim, NUM_FEATURES) and w2 has shape (1, h

    #TODO: Replace this with whatever you want to use to represent the network; yo
    model = (w1,w2)

    return model
```

## 1. Number of Neurons in Hidden Layer = 25

```
Learning Rate = 3, Iteration = 200
-------------------------------
Iteration #20: 1418.83.
Iteration #40: 1398.02.
Iteration #60: 1397.85.
Iteration #80: 1329.78.
Iteration #100: 1346.7.
Iteration #120: 1335.81.
Iteration #140: 1314.9.
Iteration #160: 1311.91.
Iteration #180: 1399.47.
Iteration #200: 1293.37.
Best model dev accuracy: 0.844625
-------------------------------
```

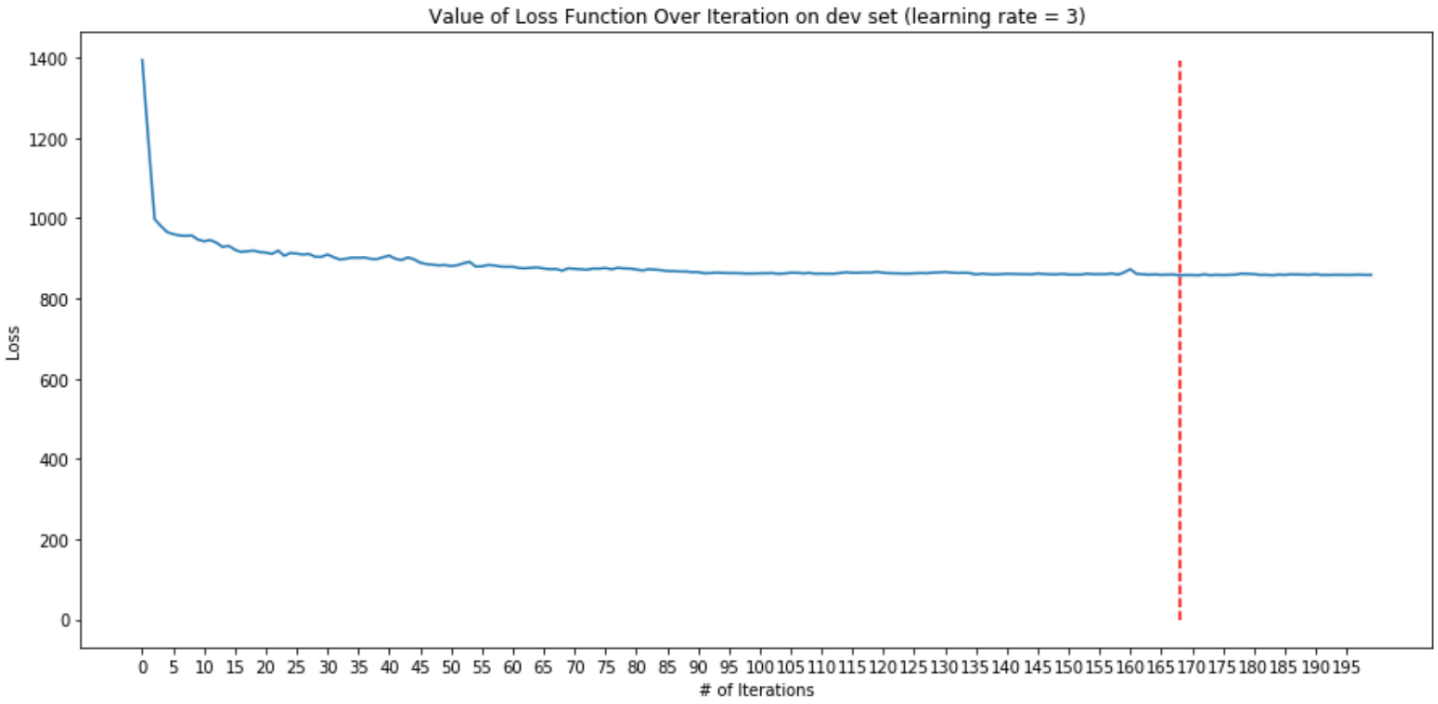



The first experiment used $learning\ rate = 3$ and $25$ neurons for the hidden layer (not including the bias term). As shown in the graph above, the value of loss function fluctuate violently and roughly converges near $200$ iterations.

So I decide to decrease the size of hidden layer and perform the next experiment.

## 2. Number of Neurons in Hidden Layer = 5

```
Learning Rate = 3, Iteration = 200
------------------------------
Iteration #20: 915.69.
Iteration #40: 902.83.
Iteration #60: 878.83.
Iteration #80: 874.48.
Iteration #100: 862.17.
Iteration #120: 866.45.
Iteration #140: 860.47.
Iteration #160: 864.84.
Iteration #180: 861.53.
Iteration #200: 858.79.
Best model dev accuracy: 0.84275
------------------------------
```



The second experiment still used $learning\ rate = 3$ but changed the number of neurons in the hidden layer to $5$ (not including the bias term). As shown in the graph above, the value of loss function is smaller and also converges much smoother than before.

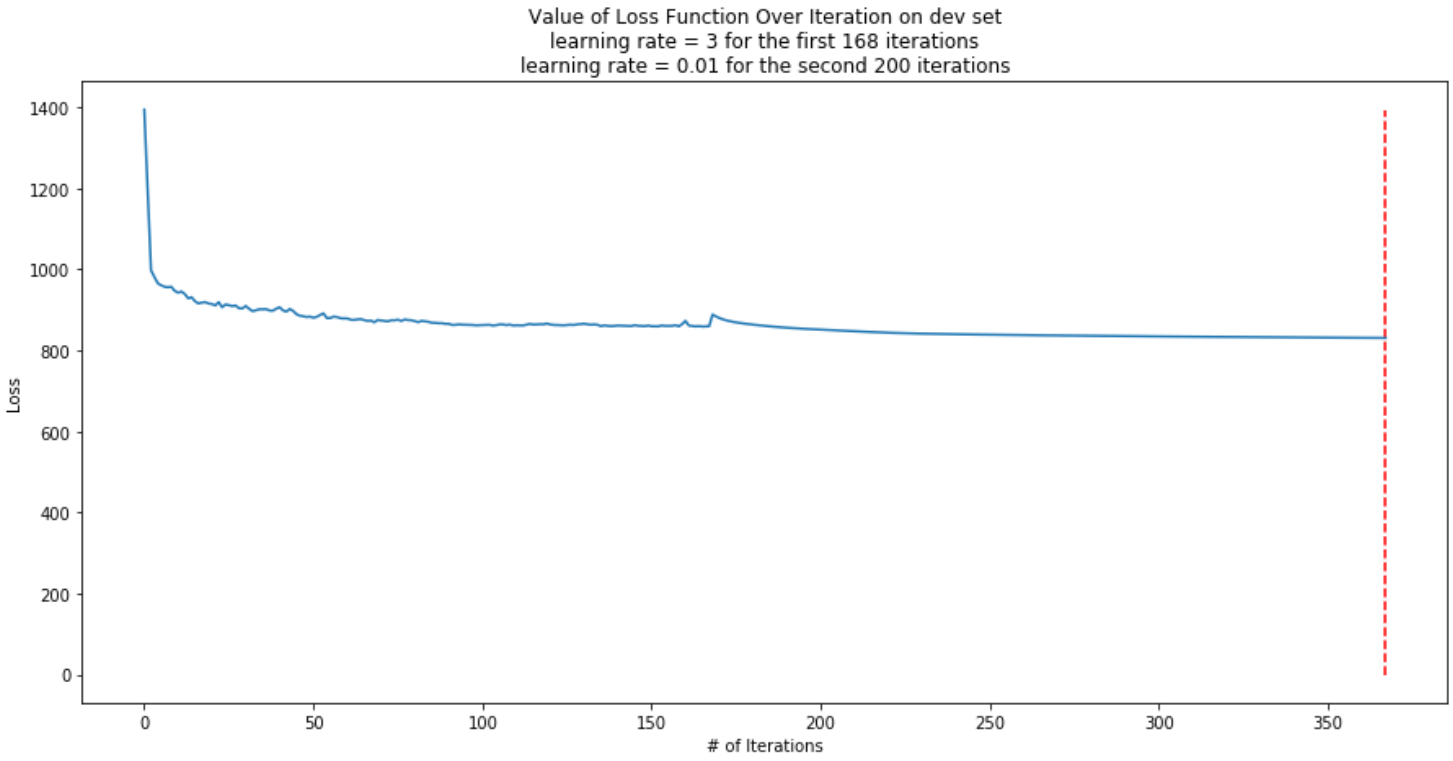
Based on the accuracy test on the `dev` dataset, the best model came from iteration #169 with a loss value of $857.53$ and accuracy of $0.84275$.

Next, I extract the model from iteration #169 and continue training with different learning rate as well as number of iterations.

```
Learning Rate = 0.01
-----------------------------
Iteration #20: 857.86.
Iteration #40: 848.59.
Iteration #60: 841.94.
Iteration #80: 839.09.
Iteration #100: 837.12.
Iteration #120: 835.47.
Iteration #140: 834.05.
Iteration #160: 832.81.
Iteration #180: 831.72.
Iteration #200: 830.75.
Best model dev accuracy: 0.845875
-----------------------------
Learning Rate = 0.5
-----------------------------
Iteration #20: 828.2.
Iteration #40: 820.51.
Iteration #60: 815.72.
Iteration #80: 812.63.
Iteration #100: 810.39.
Iteration #120: 808.58.
Iteration #140: 807.08.
Iteration #160: 806.02.
Iteration #180: 805.09.
Iteration #200: 804.12.
Best model dev accuracy: 0.84275
-----------------------------
Learning Rate = 1
-----------------------------
Iteration #20: 840.45.
Iteration #40: 831.5.
Iteration #60: 826.51.
Iteration #80: 823.6.
Iteration #100: 821.42.
Iteration #120: 819.51.
Iteration #140: 818.59.
Iteration #160: 817.36.
Iteration #180: 816.14.
```

```
Iteration #200: 815.13.
Best model dev accuracy: 0.84275
------------------------------
Learning Rate = 2
------------------------------
Iteration #20: 863.13.
Iteration #40: 859.71.
Iteration #60: 854.08.
Iteration #80: 851.27.
Iteration #100: 850.33.
Iteration #120: 848.3.
Iteration #140: 845.59.
Iteration #160: 846.0.
Iteration #180: 844.18.
Iteration #200: 841.58.
Best model dev accuracy: 0.84275
------------------------------
Experiment took 16 minutes
Best model dev accuracy: 0.84275, Learning Rate = 0.01, Iteration = 200
```

The best model from all combinations of learning rate and number of iterations is the one with $learning\ rate = 0.01$ and $iteration = 200$

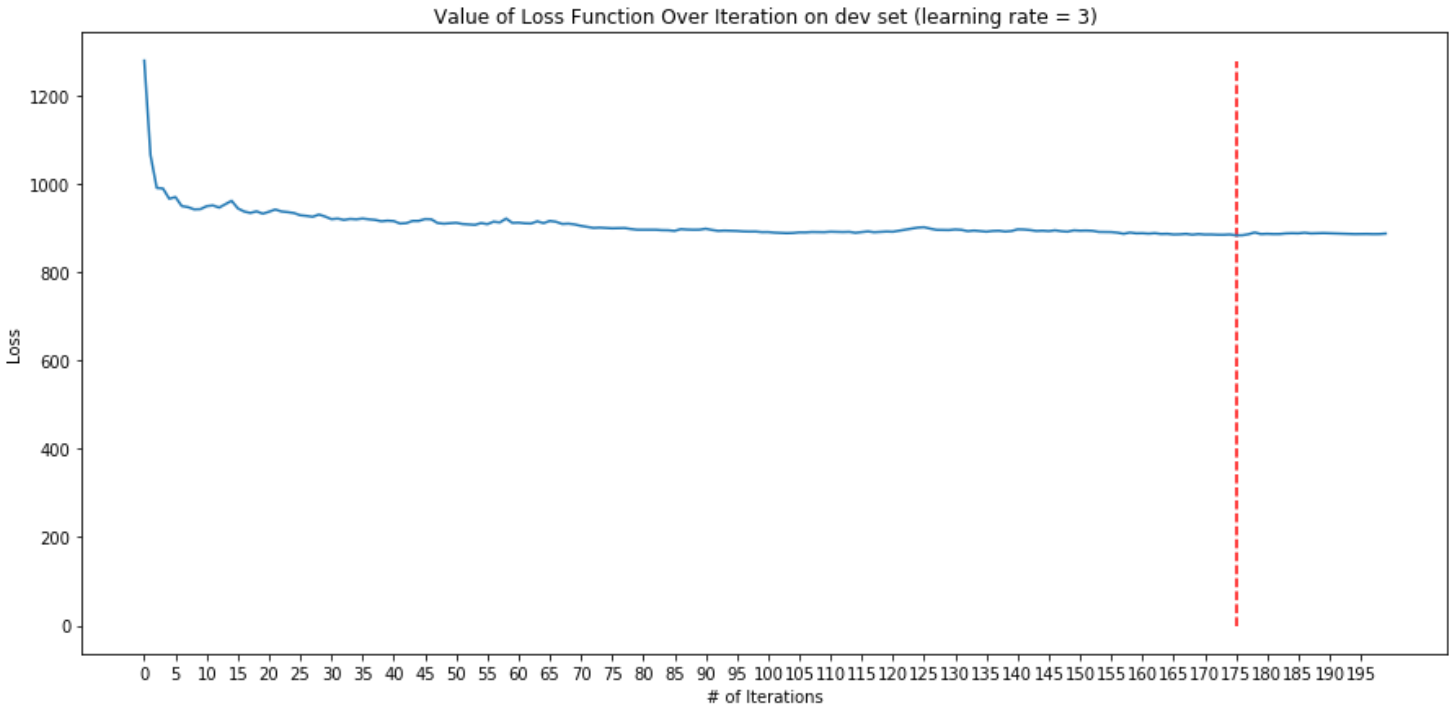



Visualizing the value of loss function from both stages, we can see after a immediate decline at the beginning, the loss value is gradually decreasing in general but seems to be able to continue decreasing if we add more iterations to the training

The final model has a loss value of $830.75$ and the accuracy of $0.84275$

### 3. Number of Neurons in Hidden Layer = 3

```
Learning Rate = 3, Iteration = 200
------------------------------
Iteration #20: 932.3.
Iteration #40: 916.55.
Iteration #60: 911.51.
Iteration #80: 896.0.
Iteration #100: 890.59.
Iteration #120: 891.92.
Iteration #140: 892.99.
Iteration #160: 887.83.
Iteration #180: 886.17.
Iteration #200: 887.31.
Best model dev accuracy: 0.849375
------------------------------
```



In the third experiment, I still used $learning\ rate = 3$ but changed the number of neurons in the hidden layer to only $3$ (not including the bias term). As shown in the graph above, the value of loss function converges at similar number of iterations.

Based on the accuracy test on the `dev` dataset, the best model came from iteration #175 with a loss value of $883.42$ and accuracy of $0.849375$.

Similar to the second experiment, I extract the model from iteration #175 and continue training with different learning rate as well as number of iterations.

```
Learning Rate = 0.01
-----------------------------
Iteration #20: 861.69.
Iteration #40: 858.41.
Iteration #60: 856.17.
Iteration #80: 854.48.
Iteration #100: 853.1.
Iteration #120: 851.93.
Iteration #140: 850.91.
Iteration #160: 850.01.
Iteration #180: 849.2.
Iteration #200: 848.46.
Best model dev accuracy: 0.849375
-----------------------------
Learning Rate = 0.5
-----------------------------
Iteration #20: 849.16.
Iteration #40: 843.28.
Iteration #60: 840.73.
Iteration #80: 838.9.
Iteration #100: 837.45.
Iteration #120: 836.24.
Iteration #140: 835.16.
Iteration #160: 834.14.
Iteration #180: 833.32.
Iteration #200: 832.65.
Best model dev accuracy: 0.849375
-----------------------------
Learning Rate = 1
-----------------------------
Iteration #20: 863.92.
Iteration #40: 859.67.
Iteration #60: 855.03.
Iteration #80: 851.72.
Iteration #100: 849.79.
Iteration #120: 848.45.
Iteration #140: 846.56.
Iteration #160: 845.62.
Iteration #180: 844.77.
Iteration #200: 844.22.
```

```
Iteration #200: 844.22.
Best model dev accuracy: 0.849375
------------------------------
Learning Rate = 2
------------------------------
Iteration #20: 893.09.
Iteration #40: 886.22.
Iteration #60: 876.76.
Iteration #80: 872.17.
Iteration #100: 867.13.
Iteration #120: 867.38.
Iteration #140: 864.98.
Iteration #160: 864.15.
Iteration #180: 864.17.
Iteration #200: 859.65.
Best model dev accuracy: 0.849375
------------------------------
Experiment took 16 minutes
Best model dev accuracy: 0.849375, Learning Rate = None, Iteration = None
```

The result turns out that additional training does not yield better model. The best accuracy on the `dev` dataset is still $0.859375$ and the best values for learning rate and number of iterations remain `None` and have not been changed.

The reason for this might be that due to a less complex model (3 neurons vs 5 neurons), the model converges faster.